Solvers for (O) (N) Electronic Structure in the Strong Scaling Limit

Bock, Nicolas
Challacombe, William M.
Kale, Laxmikant

SOLVERS FOR $\mathcal{O}(N)$ ELECTRONIC STRUCTURE IN THE STRONG SCALING LIMIT

NICOLAS BOCK , MATT CHALLACOMBE*, AND LAXMIKANT V. KALÉ†

Abstract. We present a hybrid OpenMP/Charm**++** framework for solving the $\mathcal{O}(N)$ Self-Consistent-Field eigenvalue problem with parallelism in the strong scaling regime, $P \gg N$, where P is the number of cores, and N a measure of system size, *i.e.* the number of matrix rows/columns, basis functions, atoms, molecules, *etc.* This result is achieved with a nested approach to Spectral Projection and the Sparse Approximate Matrix Multiply [Bock and Challacombe, *SIAM J. Sci. Comput.* **35** C72, 2013], and involves a recursive, task-parallel algorithm, often employed by generalized N-Body solvers, to occlusion and culling of negligible products in the case of matrices with decay. Employing classic technologies associated with generalized N-Body solvers, including over-decomposition, recursive task parallelism, orderings that preserve locality, and persistence-based load balancing, we obtain scaling beyond hundreds of cores per molecule for small water clusters ($[\mathrm{H_2O}]_N$, $N \in \{30, 90, 150\}$, $P/N \approx \{819, 273, 164\}$) and find support for an increasingly strong scalability with increasing system size N.

Key words. Sparse Approximate Matrix Multiply; Sparse Linear Algebra; SpAMM; Reduced Complexity Algorithm; Linear Scaling; Quantum Chemistry; Spectral Projection; N-Body; Charm++; Matrices with Decay; Parallel Irregular; Space Filling Curve; Persistence Load Balancing; Over-decomposition

AMS subject classifications. 65F15, 65-04, 65Z15, 15-04

1. Introduction.

Ab initio electronic structure methods for the Self-Consistent-Field (SCF) problem, involving pure density functional theory (DFT) [80, 91] or hybrid functionals that also include the Fock exchange [18], offer predictive power at low cost, finding broad utility in chemistry, biology, materials science and drug design. With conventional methods, solving the SCF eigenvalue problem significantly contributes to the total computational cost due to its steep $\mathcal{O}(N^3)$ scaling [44, 59] which in practice restricts problems to systems with $\sim 1,000$ atoms even on large computers [66, 74, 131, 72]. Recently, alternative methods which are $\mathcal{O}(N)$ (linear scaling) have been developed that exploit the local quantum nature of non-metallic electronic interactions. Early approaches to linear scaling solutions of the SCF eigenproblem sought to exploit this quantum locality by avoiding the pair-wise support of local basis functions beyond a cutoff radius, leading to matrix sparsity and an $\mathcal{O}(N)$ computational effort through iterative algorithms based on the sparse matrix-matrix multiply (SpMM) [94, 34, 35, 128, 77]. Later, incomplete/inexact methods based on the dropping of small elements (radial cutoffs/filtering) were developed [94, 61, 106, 47]. While current linear scaling methods can access systems involving $\sim 1,000,000$ atoms [32, 109, 133], they have yet to enjoy widespread scientific use at scale, perhaps because the demands of configurational sampling likewise increase with system size. Thus, parallel algorithms that reduce the time to solution *per atom* are key in unlocking the scientific potential of $\mathcal{O}(N)$ methods. For an excellent review and current state of the art see Bowler *et al.* [32, 33, 36].

A parallel SpMM implementation was first mentioned by Goringe *et al.* as part of the CONQUEST code using a one dimensional, row-wise matrix decomposition, although details were not reported [69]. Later, one of us introduced the distributed

*Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87544, USA, nicolas-bock@freeon.org

†Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign

blocked compressed sparse row (DBCSR) data format and corresponding algorithm for distributed sparse matrix multiplication, with space filling curve ordering and a one dimensional, row-wise matrix decomposition based on the greedy bin packing problem, demonstrating parallel efficiency of the SCF eigenproblem up to 128 cores [47]. More recently, space filling curve ordering schemes to improve locality and data layout in radial cutoff schemes [34, 37] and two-dimensional matrix decompositions [30] have lead to improved efficiencies. Bowler *et al.* reported a scalable SpMM on 196,000 cores involving $\sim 1,000,000$ atoms [32, 109], while VandeVondele *et al.* demonstrated scalability of $\sim 1,000,000$ atoms on 46,656 cores [133]. Also, generic methods for the SpMM have been developed by Buluç *et al.* where matrix row and columns are randomly permuted to achieve an even load distribution, yielding high efficiencies [38, 39, 42, 41, 40]. This approach has been adopted for quantum chemistry with a slightly modified Cannon algorithm [45], radial cutoffs, and static load balancing based on a fixed graph [30].

These parallel approaches to the $\mathcal{O}(N)$ SCF eigenvalue problem, based on one- or two-dimensional strategies for matrix decomposition, have established scalability in the weak regime, $P/N \approx$ constant, where P is the number of cores and N is the system size (see for example Fig. 1 of Ref. [36]). However, bounding communication costs to achieve scalability beyond the weak regime remains challenging [17] and will gain in importance for the increasingly large, asynchronous, and heterogeneous next generation of high performance computing systems with $P > 1,000,000$ cores[1]. In addition, current randomization strategies [38, 133] that forgo locality are throttled to $\mathcal{O}(\log P)$ [17] due to the cost of their communication algorithms, *e.g.* SUMMA [132]; lowering these communication costs will require either an *a priori* knowledge of sparsity patterns, or pre-computing and packing of non-zero elements before communication [17]. So far, even prototypes of either strategy have yet to appear.

Recently we have developed an N-Body approach to the linear algebra of data-local matrices with decay, involving the recursive occlusion of sub-multiplicative norms based on the Cauchy-Schwarz inequality [50, 28, 27]. Besides wide application in physical simulation [137, 139, 140, 90, 82, 118, 75, 68, 111, 136, 135], N-Body methods find broad applicability in statistical learning [71, 120, 110, 93, 92] and database operations [108, 79]. Our Sparse Approximate Matrix Multiply (SpAMM) algorithm is loosely comparable to the solution of Poisson's equation through N-Body simulation with radial cutoff, which has been shown recently to exhibit communication optimal bounds, $\mathcal{O}(1/P)$, for locality preserving spatial decompositions [64]. With heuristic schemes that parlay quantum locality into spatial and temporal data locality, together with persistence based load-balancing and three-dimensional over-decomposition strategies, the communication cost of SpAMM may be limited in a similar fashion.

In modern electronic structure theory there are typically four additional "fast" solvers beyond the SCF eigenproblem that must interoperate with each other, representing a tightly coupled collective of advanced numerical methods. Historically, these solvers have been developed and optimized independently, involving differing data structures and programming models (*e.g.* 3-D FFT, CSR based SpMM, transformations of basis function to numerical grids *etc.*). In the strong scaling regime, such a piecemeal collection may: (a) disrupt data locality with redistributions and transformations, (b) significantly raise the barrier to entry and innovation, (c) exceed

[1]Already, the number of cores in the current top 5 supercomputers is close to or even exceeds this number: Tianhe-2 – 3,120,000 cores, Titan – 560,640 cores, Sequoia – 1,572,864 cores, K – 705,024 cores.

the ability of advanced runtime systems to load balance multiple programming models, (d) lead to divergent rates of error accumulation, and (e) impede deployment for trends such as fine grained check-pointing [119, 147, 127], fault-tolerance [134], energy aware load balancing [98, 125] and job malleability [88, 124].

We have recast all five solvers at the hybrid HF/DFT level of SCF theory within the generalized N-Body solvers framework, including (1) Fock exchange [51], (2) spectral projection (this work), (3) inverse factorization [52], (4) Coulomb summation [53, 54, 55, 56] and (5) the exchange correlation problem [48]. These developments offer a potential solution to challenges (a)-(e), through a unified approach with a proven record of performance [137, 139, 140, 90, 82, 118, 75, 68, 111, 136, 135]. In this contribution, we develop strategies for recursive over-decomposition and persistence-based load balancing of the SpAMM kernel [50, 28] as employed by spectral projection, an $\mathcal{O}(N)$ alternative to the SCF eigenvalue problem for matrices with decay [112]. Ultimately, generic N-Body frameworks and associated parallelization strategies, explored here in part, may lead to broad horizontal support and cohesion across entire solver collectives, enabling access to the strong scaling regime for complex problems such as electronic structure.

It should be pointed out that the density matrix constructed through purification schemes, such as the method of Palser and Manolopoulos [116] and the SP2 method [112] do not retain contact with the Hamiltonian eigenspace, exponentially accumulating numerical errors under inexact/incomplete approximation [113]. In addition, spectral projection solvers can not be preconditioned with the density matrix from a previous step, *e.g.* within a molecular dynamics or structure optimization procedure, negatively impacting overall performance [116]. On the other hand, variational approaches such as the methods of Li, Nunes, Vanderbilt [94], and Daw [61] retain contact with the eigenspace of the Hamiltonian through the gradient [31], however, convergence can be very slow. More recently Newton-Schulz techniques have been developed which yield accelerated rates of convergence, and maintain direct contact with the Hamiltonian eigenspace [58, 52].

This paper is organized as follows: In Sec. 2 we describe in detail the SpAMM algorithm and in Sec. 3 its parallel implementations within OpenMP and the Charm++ runtime. In Sec. 4 we detail our methodology and show parallel scaling results for quantum mechanical matrices with decay and demonstrate scalable high-performance in the strong scaling limit. Finally, we discuss our results in Sec. 5.

2. The Sparse Approximate Matrix Multiply. A wide class of problems exist that involve matrices with decay, often corresponding to matrix functions [21], notably the matrix inverse [63, 23], the matrix exponential [81], and in the case of electronic structure theory, the Heaviside step function (spectral projector) [103, 116, 46, 47, 22, 19]. A matrix A is said to decay when its matrix elements decrease exponentially, as $|a_{ij}| < c\,\lambda^{|i-j|}$, or algebraically as $|a_{ij}| < c/(|i-j|^{\lambda}+1)$ with separation $|i-j|$. See Benzi for an excellent discussion [21, 23, 22, 20]. In simple cases, the separation $|i-j|$ may correspond to an underlying physical distance $|\vec{r}_i - \vec{r}_j|$, *e.g.* of basis functions, finite elements, *etc.* [19], leading often to a strong diagonal dominance when ordered carefully [47]. For simple decay, truncation in the two-dimensional vector space, *e.g.* via radial cutoff $a_{ij} = 0$ if $|\vec{r}_i - \vec{r}_j| > r_{\text{cut}}$ [31], a numerical threshold $a_{ij} = 0$ if $|a_{ij}| < \epsilon$ [47], or by restricting matrix operations to a known sparsity pattern [30], together with the use of a conventional SpMM algorithm [73], yields a reduced complexity kernel for the iterative construction of matrix functions. When matrix operations are restricted to a known sparsity pattern, the matrices retain their sparsity by construction throughout the iterative process. But when radial or numerical truncation schemes are employed the matrices will fill-in unless repeatedly filtered [33].

Truncation may not be the most efficient or accurate approach to exploiting decay, which can be oscillatory, involve quantum beats, or even long range charge transfer as in the case of excited states, see for example Ref. [49] and references therein. In addition, exploiting the secondary "lensing" effects in higher dimensional operation spaces within truncation schemes is challenging [52]. These effects are shown in Fig. 2.1, which shows a density matrix for a large water cluster with the underlying basis ordered to preserve locality; note the large *anti-diagonal* beats, as well as the strong clustering and segregation of elements with like magnitude. For this type of structured matrix with non-trivial decay, the quadtree [122, 123, 141, 142]

$$A^t = \begin{pmatrix} A_{11}^{t+1} & A_{12}^{t+1} \\ A_{21}^{t+1} & A_{22}^{t+1} \end{pmatrix}, \tag{2.1}$$

where t denotes the tier, pioneered in linear algebra by Wise *et. al* [143, 14, 15, 65, 70, 97, 144], provides a powerful framework for recursive database operations such as the *metric-query* [16, 79, 83], involving the lookup of sub-blocks by magnitude, $\|A_{ij}^t\|$. In this work we use the Frobenius norm, which is cheap to hierarchically compute from submatrix norms,

$$\|A\| = \sqrt{\sum_{ij} |A_{ij}|^2} \tag{2.2}$$

and

$$\|A^t\| = \sqrt{\sum_{i,j=1}^{2} \|A_{ij}^{t+1}\|^2}. \tag{2.3}$$

Based on this framework, the SpAMM algorithm [50, 28]

$$C_{ij}^t \leftarrow C_{ij}^t + \sum_{k=1}^{2} \begin{cases} A_{ik}^t B_{kj}^t & \text{for } \|A_{ik}^t\|\|B_{kj}^t\| > \tau \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

exploits decay recursively in the three-dimensional convolution space, with adaptive culling and occlusion of insignificant products at each tier t, determined by application of the sub-multiplicative norm inequality, $\|A\,B\| \leq \|A\|\|B\|$, and a numerical threshold τ controlling precision.

While the discussion has so far involved dense matrices, SpAMM is applicable to sparse matrices as well. Also, even with dense matrices, large values of τ correspond to an implicit truncation and potentially a sparse product. Relative to conventional row-column approaches to the SpMM, the SpAMM algorithm applied to structured, data-local matrices with decay may achieve: (*i*) additional flexibility in the three-dimensional task space for domain decomposition and load balancing (this work), (*ii*) the recursive accumulation of terms with like magnitude and an $\mathcal{O}(N \lg N)$ error accumulation [28, 25], (*iii*) occlusions that occur early in recursion enabling communication optimal approaches, (*iv*) a more efficient use of high level memory chunking for message passing and low level blocking strategies for acceleration (also this work), and (*v*) additional flexibility for achieving error control within a culled volume, and complexity reduction via lensing.

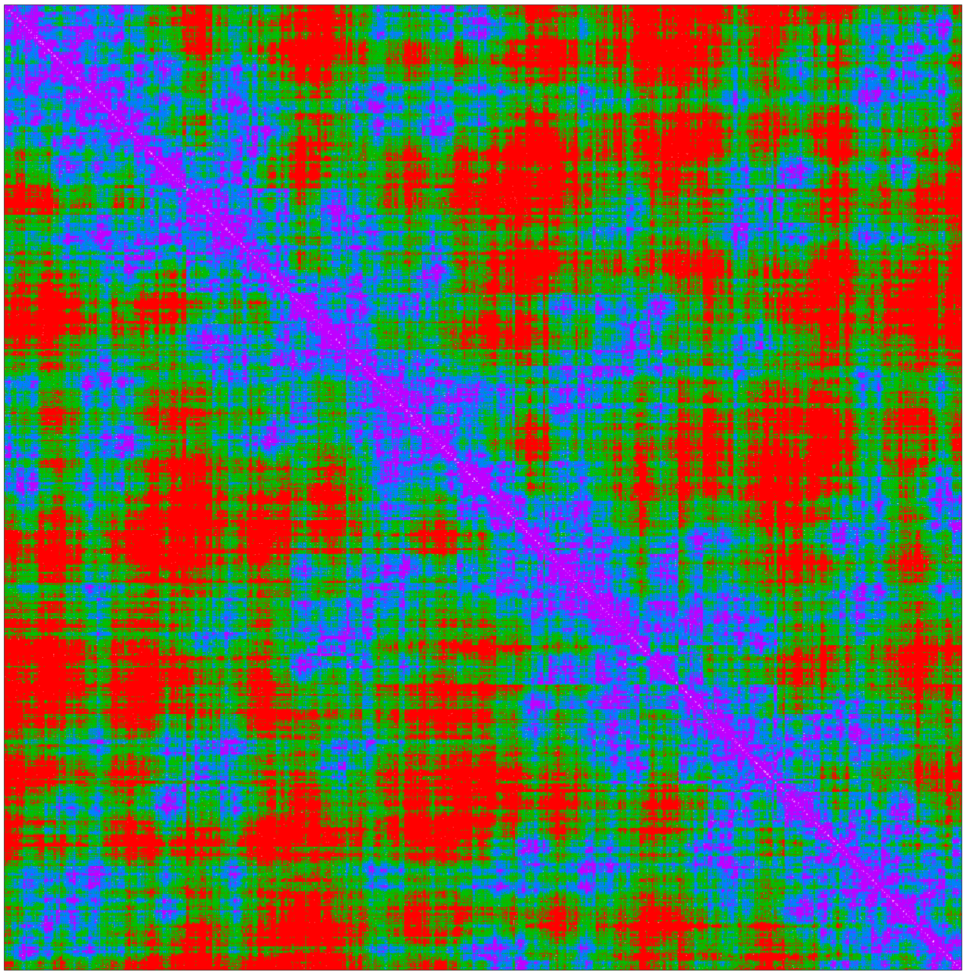


Fig. 2.1: The decay of matrix element magnitudes of a converged spectral projector (density matrix) for a $(H_2O)_{300}$ water cluster at the RHF/6-31G** level of theory ($n = 7500$), where the molecular geometry has been reordered with a space filling Hilbert curve. The different colors indicate different matrix element magnitudes; red: $\left[0, 10^{-8}\right)$; green: $\left[10^{-8}, 10^{-6}\right)$; blue: $\left[10^{-6}, 10^{-2}\right)$; violet: $\left[10^{-2}, 1\right]$, corresponding to approximate exponential decay.

3. Task Over-Decomposition. One of the strengths of the generalized N-Body framework is that there are many ways to realize over-decomposition on a range of hardware, *e.g.* from long pipe GRAPE single instruction, multiple data (SIMD) accelerators [111, 101, 100, 99, 89] to conventional symmetric mulitprocessing (SMP) and multiple instruction, multiple data (MIMD) architectures [79, 95, 135, 139, 138, 137, 140, 68, 75, 82, 90, 118]. Ideally, an architecture independent runtime system seamlessly enables the recursive generation of lightweight tasks, as OpenMP 3.0 does for SMP. However, while this feature is a target of the Dynamic Parallelism framework of NVIDIA's CUDA 5.0 [4] and at least partially included in a number of parallel runtimes such as Intel's Threading Building Blocks (TBB) [6], Concurrent Collections (*e.g.* Intel's CnC [3]), Wool [13], Nanos++ [7], OpenUH [11], Intel's Cilk Plus [5], the Open Community Runtime (OCR) [8], OpenCL [9], TASCEL [12], *etc.*, full support for recursive task parallelism is mostly unrealized for distributed memory systems at present. In this work, we consider simple methods for achieving recursive task parallelism with SpAMM for the ubiquitous "cluster of SMP nodes" architecture [129, 76, 145, 62] using two runtimes, OpenMP and Charm++, within a hybrid approach.

There are two main considerations in our scheme that involve memory and task management: First, the naïve use of task parallelism at the SMP level, with either OpenMP or Charm++, has the potential to involve non-contiguous memory and high packing/unpacking overheads when redistributing memory between nodes, potentially negatively impacting overall performance. Additionally, the cache hierarchy of modern CPUs with small, local caches and large, shared last level caches should not be ignored. Second, an explicitly allocated, unrolled octree is a necessary structure that enables Charm++ to manage tasks involving occlusion and culling as well as node-level SMP work due to limitations of the load-balancing framework implemented in Charm++.

Thus, we allocate contiguous chunks of size $N_c \times N_c$ to hold a full sub-quadtree together with a $N_b \times N_b$ blocking at the lowest level. The chunks are processed using OpenMP and the code can potentially be used without modification on the Intel Xeon Phi coprocessor and through automatic source code translation [114, 102, 121, 57] on GPGPUs. In addition, the use of OpenMP removes Charm++ compile and runtime dependencies for single-node applications, potentially significantly simplifying the build process.

We expect the overall performance to be influenced by several competing size-dependent effects: (1) The ratios N/N_c and N_c/N_b limit the maximum number of tasks available for load-balancing for Charm++ and OpenMP respectively, and (2) the leaf node size N_b affects the performance of memory access through the CPU's cache hierarchy and the potential for vectorization and convolution space compression. While we previously demonstrated that a highly specialized and optimized dense kernel can lead to competitive performance for very small dense submatrices of $N_b = 4$ [28], the use of manually tuned assembly code renders this approach rigid with respect to submatrix granularity and width of SIMD vectors. Thus, in this work, we implemented a simple kernel with three nested loops and leave low-level optimizations to the compiler.

3.1. OpenMP. Shown in Alg. 1 is the SMP parallel implementation within the OpenMP application programming interface; `SpAMM_omp` recursively walks a transient octree generated dynamically on the stack through the OpenMP 3.0 tasking feature [10]. Guided by the binary convolution of matrix quadtrees A^t and B^t at each tier t (line 3), the implicit octree traversal may be sparse and irregular due to culling and occlusion (line 4) based on the sub-multiplicative matrix norm inequality, $\|A\,B\| \leq \|A\|\|B\|$. The parallel tree traversal is extended through untied OpenMP `tasks` (line 5) and recursive calls to `SpAMM_omp` (line 6). Per node synchronization (to ensure appropriate variable lifetimes) is achieved through the OpenMP `taskwait` statement (line 9). Finally, at the leaf tier, dense matrix products are performed and the result reduced into the C quadtree (line 12), with a data race on C prevented through explicit use of OpenMP locks (lines 11, 13). While other approaches to address data write contention are certainly possible, *e.g.* OpenMP reductions or atomics, we found good on-node parallel scaling using explicit locks, and defer such potentially performance enhancing details to a forthcoming article.

In this work we have made only modest effort to optimize the `SpAMM_omp` implementation, or even the dense contraction on line 12. In Ref. [28] we showed that accuracies better than the native GEMM are possible also with N-scaling, but that difficult, platform specific optimizations were necessary; we are currently developing a corresponding OpenMP algorithm and are investigating the use of compiler vectorization and OpenMP 4.0 SIMD constructs [10].

Algorithm 1 The OpenMP SpAMM algorithm, recursively multiplying matrices $C \leftarrow A \times B$ under a SpAMM tolerance τ. The function matrix arguments are pointers to tree nodes.

```
1: function SpAMM_omp($\tau$, $t$, $A^t$, $B^t$, $C^t$)
2:     if $t <$ depth then
3:         for all $\left\{ i, j, k \mid C^t_{ij} \leftarrow A^t_{ik} B^t_{kj} \right\}$ do
4:             if $\|A_{ik}\| \, \|B_{kj}\| > \tau$ then                        ▷ Culling
5:                 OpenMP task untied
6:                 SpAMM_omp($\tau$, t+1, $A^{t+1}_{ik}$, $B^{t+1}_{kj}$, $C^{t+1}_{ij}$)
7:             end if
8:         end for
9:         OpenMP taskwait
10:    else
11:        omp_set_lock                           ▷ Acquire OpenMP lock on $C$
12:        $C \leftarrow C + A \times B$                          ▷ Dense product
13:        omp_unset_lock                         ▷ Release OpenMP lock on $C$
14:    end if
15: end function
```

3.2. Charm++. Charm++ [2] is a mature runtime environment on distributed memory platforms available for all major supercomputer systems, allowing for efficient scalable high performance implementations [105, 125, 86, 96, 85, 67, 104, 87, 24, 146]. In the message-driven execution model of Charm++, code and data are encapsulated in C++ objects called "chares" which are initially placed by static load balancing algorithms. Dynamic persistence-based load balancing strategies migrate chares transparently during solver execution based on load and communication measurements from previous solver iterations and efficiently optimize load distribution and communication cost. The Charm++ runtime transparently manages chare placement and migration and proxy objects are used to send messages to particular chare instances or groups thereof without explicit specification of their location. Chares can be grouped in multi-dimensional sparse arrays or used as "singleton" objects.

Persistence-based load balancing exploits temporal and spatial localities in iterative solvers through decomposition of the load and communication graph. Since the dynamic load balancing strategies of Charm++ only consider chares organized in arrays persistently instantiated across solver operations and load balancing, the multiplication octree has to be explicitly stored in memory (as opposed to the transient stack based "storage" used in the SMP implementation). Note that such persistent allocation of the multiplication octree could aid efficient load balancing across molecular dynamics or structure optimization steps, see *e.g.* the impressive scaling of astrophysics applications [67, 84, 135]. The nodes of the matrix quadtree between root, $t = 0$, and chunks, $t = t_c$, given by N_c, are stored in a stack of two-dimensional chare arrays of size $2^t \times 2^t$ each. The corresponding unrolled octree is stored in three-dimensional chare arrays with occlusion and culling carried out iteratively, tier-by-tier, until the chunk level at which `SpAMM_omp` is invoked.

Data and work locality are exploited through the communication aware load balancing strategies in Charm++. However, at the time of this writing, a bug in the Charm++ runtime [26] prevents the use of *sparse* load balanced chare arrays. As a work-around, we mark chares that correspond to pruned tree nodes with a boolean data member, `isDisabled == true`, introducing a $\mathcal{O}(N^3)$ communication component with a prefactor found to be negligible.

Algorithm 2 The SpAMM algorithm in the Charm++ programming language. Tree occlusion is done by iterating over the three-dimensional multiplication chare arrays, CONVOLUTION$[d]$. In Charm++ a call such as CONVOLUTION$[t]$.OCCLUDE translates into a broadcast to all array elements of CONVOLUTION$[t]$.

```
1: function SpAMM_charm($\tau$, $A$, $B$, $C$)
2:     for $t \geq 0 \wedge t < d$ do
3:         CONVOLUTION[$t$].OCCLUDE($\tau$)                ▷ See Alg. 3
4:     end for
5:     CONVOLUTION[$d$].MULTIPLY
6:     CONVOLUTION[$d$].STORE
7: end function
```

The Charm++ algorithm is outlined in Alg. 2 and proceeds in three phases. In the first phase, the multiplication octree is constructed iteratively over the top tiers of the three-dimensional chare arrays, shown in lines 2 and 3 of Alg. 2. This phase is a breadth-first implementation of the SpAMM algorithm and retains the full complexity reduction of the depth-first, recursive implementation, Alg. 1. In each iteration of this phase, a broadcast message is sent to all multiplication chares of tier t (line 3) executing the OCCLUDE method on the enabled array elements, shown in Alg. 3. The scalar products of the eight matrix norms of the A and B nodes of the next tier are formed, lines 6-10 of Alg. 3, and Eq. 2.4 is used to decided whether to enable or disable the corresponding multiplication chares. Disabled multiply chares (`isDisabled == true`) are skipped during the next iteration of the pruning phase, shown in lines 2-4 of Alg. 2.

During the second phase, line 5 of Alg. 2, the SMP SpAMM code is called to compute the $N_c \times N_c$ submatrix products in each remaining, enabled multiplication chare, and the results are stored in a temporary variable local to the chare. In the final phase, line 6 of Alg. 2, all temporary matrix products are gathered in the STORE method, summed, and added to the corresponding chares of C. Since the Charm++ runtime guarantees exclusive execution of chare instances, explicit locking or other means of synchronization as in the OpenMP implementation are not necessary.

3.3. The OpenMP/Charm++ Hybrid. In our hybrid approach, we found the best performance with one Charm++ Processing-Element (PE) per node, OpenMP commanding all on-node threads and $N_c \times N_c$ quadtree chunking as discussed above. This approach avoids the problem of packing and unpacking fragmented memory during chare migration, enabling use of a single `memcpy`, which is efficient in standard libraries such as `libc`. Certainly, optimal chunk and block sizes are likely to be application dependent, an issue beyond the scope of the current work. A further complication of the hybrid approach involves the issue of local *vs.* absolute addressing;

Algorithm 3 Tree occlusion in the Charm++ programming language of the multiplication chare element on tier t with index (i, j, k). In Charm++ the call CONVOLUTION$[t+1](i, j, k)$.ENABLE translates into a direct message to the ENABLE method of the multiplication chare element with index (i, j, k) on tier $t+1$.

```
function OCCLUDE($\tau$)                                    ▷ On tier $t$, index $(i, j, k) \in [1, 2^t]$
    if isDisabled then
        return
    end if
    for all $\left\{ i', j', k' \,\middle|\, C^{t+1}_{i',j'} \leftarrow A^{t+1}_{i'k'} B^{t+1}_{k'j'} \right\}$ do
        if $\|A^{t+1}_{i'k'}\| \|B^{t+1}_{k'j'}\| > \tau$ then
            CONVOLUTION$[t+1](i', j', k')$.ENABLE
        else
            CONVOLUTION$[t+1](i', j', k')$.DISABLE
        end if
    end for
end function
```

by wrapping an address offset with convenience macros, the OpenMP application programming interface given in Alg. 1 can be used without modification.

4. Results.

In this work we consider scalability of the SpAMM kernel in the context of spectral projection [103, 116, 46, 47, 22, 19], an alternative to explicitly solving the SCF eigenvalue problem [130]. Spectral projection involves nested construction of the matrix Heaviside step-function from the effective SCF Hamiltonian (Fockian), in our case computed in a basis of atom-centered functions [130]. In this work, tightly converged, dense matrices for a sequence of water clusters were computed at the B3LYP/6-31G** level of theory [18] using `FreeON`, a suite of programs for $\mathcal{O}(N)$ quantum chemistry [29]. This sequence of water clusters corresponds to standard temperature and pressure, and has been used in a number of previous studies [56, 55, 53, 43, 126, 107, 60, 115, 28]. The 6-31G** basis set introduces 5 basis function per hydrogen atom and 15 basis functions per oxygen atom, yielding 25 basis functions (and matrix rows and columns) per water molecule. A key aspect of this work is ordering of the atom indices with the locality preserving Hilbert curve, Ref. [47] and references therein, yielding clustering and segregation of elements by magnitude as in Fig. 2.1.

In a previous study [28], we reported linear scaling computational complexities and SpAMM errors as the max norm of the difference between the SpAMM product and a dense reference product. Here, we consider SpAMM errors that accumulate in iterative application of the second order spectral projection scheme (SP2) [112]. In all cases, the SP2 solver was run to convergence, taking 40 iterations. Values of $\tau = 10^{-6}, 10^{-8}$, and 10^{-10} are considered for scaling experiments, with $\tau = 10^{-6}$ corresponding to extreme truncation (a highly sparse representation).

All OpenMP scaling studies were run on a fully allocated 48-core, 4-socket AMD Opteron 6168 (Magny Cours architecture) system running at 1.9 GHz using `GNU gcc` 4.6.3, and a 24 core, 2-socket AMD Opteron 6176 (Magny Cours architecture) system running at 2.3 GHz using `GNU gcc` 4.7.2. The Charm++ scaling studies were run on the largest open computer cluster at Los Alamos National Laboratory (LANL), "mustang", which consists of 1,600 dual socket AMD Opteron 6176 (Magny Cours)

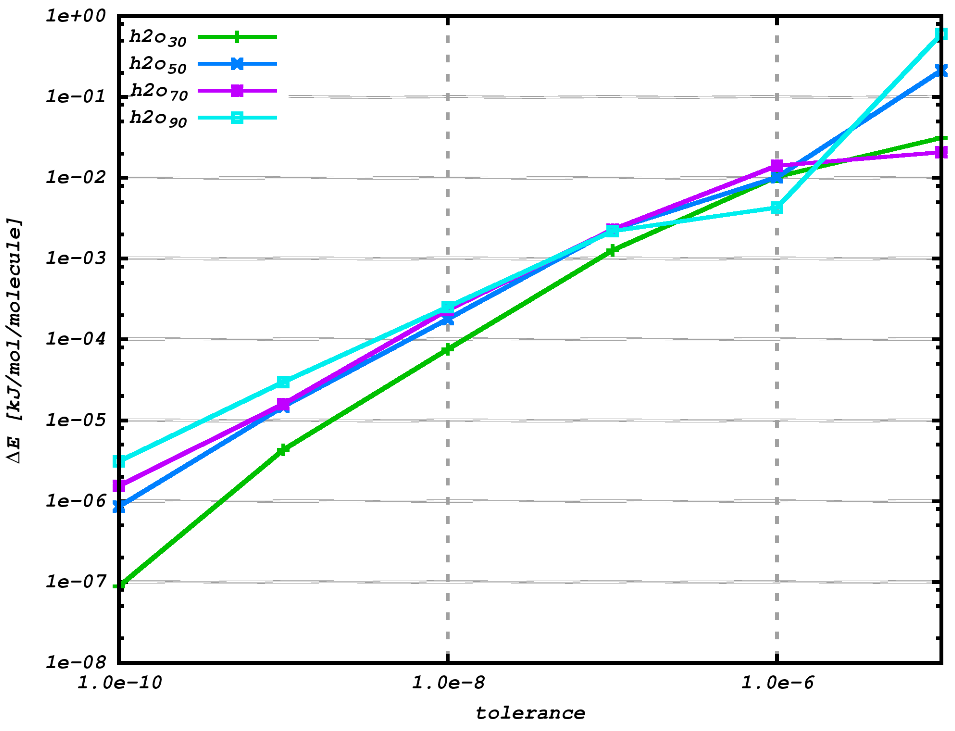
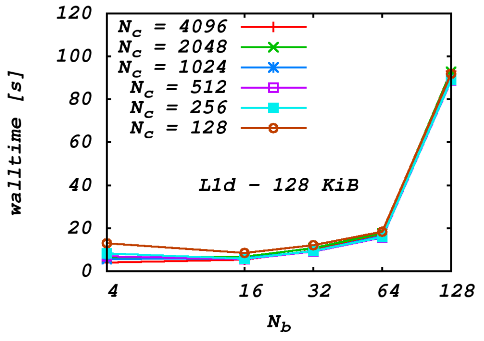


Fig. 4.1: The absolute error of the energy after 40 iterations of the spectral projection method for different water clusters in B3LYP/6-31G**.

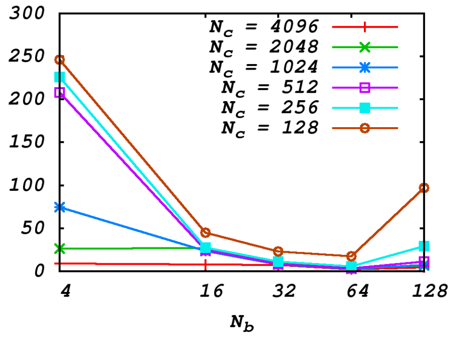
nodes for a total of 38,400 cores using `GNU gcc` 4.7.2. All tests used the `-O2` level of compiler optimization.

4.1. Error Accumulation. The accumulation of error in spectral projection due to the SpAMM kernel is computed here as $\mathrm{Tr}[F\,(P-\tilde{P})]$, where F is the Fockian, $\tilde{P}$ is the approximate density matrix computed from F with $\tau \neq 0$, and P is a reference computed with $\tau = 0$. These errors are reported in Fig. 4.1, demonstrating that the error per molecule exhibits no significant system size dependence for the cases studied here, in agreement with our earlier results on the max norm error behavior of SpAMM, Figs. 5.2 and 5.3 of Ref. [28]. Roughly, these results suggest that chemical accuracy (1 kcal/mol or 4.184 kJ/mol [78]) may be retained with 10^5 water molecules and a SpAMM threshold of $\tau = 10^{-10}$. As discussed in Sec. 2, the control of accumulated errors in the spectral projection solver is challenging due to the non-variational nature of the solver. However, our results indicate good error control even under extreme truncation conditions ($\tau = 10^{-6}$) due to the recursive occlusion and culling based on the sub-multiplicative norm inequality, as opposed to matrix element truncation directly in the vector space.

4.2. OpenMP scaling. `SpAMM_omp` (Alg. 1) was benchmarked for the last SP2 iteration of the $(H_2O)_{90}$ and $(H_2O)_{150}$ density matrices with a loose SpAMM tolerance of $\tau = 10^{-6}$. These examples are well within the linear scaling regime for SpAMM calculations [28], yet small enough to probe a lower molecule/core ratio available

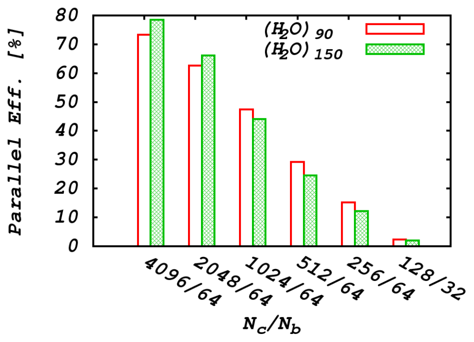


(a) SpAMM_omp restricted to 1 thread.

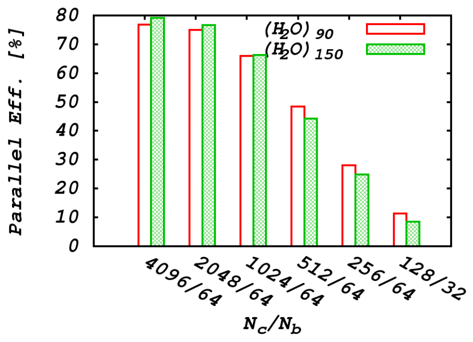


(b) SpAMM_omp on all 48 threads.

Fig. 4.2: Total time of matrix product for $(H_2O)_{90}$ of SpAMM_omp with $\tau = 10^{-6}$ restricted to one thread, (a), and on 48 threads, (b), on the Opteron 6168. In serial, we find the measured walltime to be independent of N_c, but to depend strongly on N_b. Note that $N_b > 64$ exceeds L1d leading to significant performance loss. An increase in compression due to smaller granularity leads to decreasing walltime with decreasing N_b, and an optimal block size of $N_b = 16$. Tests with $\tau = 0$ indicate that the optimal block size in serial without compression is $N_b = 64$. On 48 threads, the shortest walltime shifts from $N_b = 16$ to $N_b = 64$, shown in (b), which indicates poor memory access performance and is potentially due to a lack of cache/core affinity. It is worth noting that we find a large spread in performance for $N_b = 4$ across the chunk sizes tested.



(a) 48-core Opteron 6168.



(b) 24-core Opteron 6176.

Fig. 4.3: Parallel efficiency of OpenMP code with different granularities for B3LYP/6-31G** $(H_2O)_{90}$ and $(H_2O)_{150}$ with $\tau = 10^{-6}$. The best performing combination of N_c/N_b at the maximum number of threads ($P = 48$) was chosen for each value of N_c tested. Note that the native matrix size of $2,250 \times 2,250$ is padded to $4,096 \times 4,096$. There is little difference between the two water clusters. While larger chunks exhibit good parallel efficiency, we find a significant drop of parallel scaling for the smaller chunks.

on modern SMP platforms. In addition, such a loose SpAMM tolerance leads to irregular work loads and data access, potentially challenging the OpenMP runtime. Shown in Fig. 4.2 are walltimes for the smaller water cluster, $(H_2O)_{90}$, under several combinations of N_b and N_c in serial (OpenMP restricted to one thread), (a), and on 48 threads, (b). We find the overall performance and parallel scaling to be significantly influenced by N_c and N_b. While the performance in serial, Fig. 4.2 (a), is mostly independent of the tested chunk sizes, $N_c = \{128, 256, 512, 1024, 2048, 4096\}$, it is significantly impacted by the size of N_b. For large values, $N_b > 64$, we note a steep rise in walltime which we attribute to a lack of temporal locality in cache due to the size of per core L1d of 128KiB. The improving performance with decreasing block size, $N_b \leq 64$, is due to increasing compression in convolution space due to finer granularity. Tests with $\tau = 0$ indicate that the optimal block size in serial is $N_b = 64$. However, on 48 cores we find that the shortest walltime shifts from $N_b = 16$ to $N_b = 64$, which indicates poor memory access performance and might be due to a lack of thread/data affinity. Also, on 48 threads we find the walltime to depend more strongly on N_c, which we attribute to a lack of potential tasks for shallow trees, indicated by small ratios N_c/N_b.

In addition to walltime, it is instructive to investigate the parallel efficiency of `SpAMM_omp`. For each value of N_c we chose the value of N_b, yielding the fastest (lowest walltime) performance at 48 threads, and calculated the parallel efficiency as $E(P) = T(1)/(P\ T(P))$, where P denotes the number of threads, shown in Fig. 4.3. We note that efficiencies up to 80% can be achieved for large chunks. As N_c/N_b and the number of potential tasks decreases ($4096/64 \rightarrow 262,144$ potential tasks, $2048/64 \rightarrow 32,768$ potential tasks, $\ldots$) load balancing becomes increasingly challenging with a significant decline in parallel efficiency. Compared to $(H_2O)_{90}$, the larger water cluster, $(H_2O)_{150}$, exhibits a slightly superior parallel scaling for the larger chunks. However, the qualitative behavior with decreasing chunk size remains the same.

4.3. Charm++ Scaling. This study involved scaling with the progression $P = 24 \times 2^m$, up to 24,576 cores (1024 nodes) on LANL's largest open computer cluster "mustang". The study consisted of spectral projection via the SP2 method until convergence (40 iterations) with $m = 1, 2, \ldots, 10$, and $\tau = 10^{-6}, 10^{-8}$, and 10^{-10}. The initial data distribution during the first SP2 iteration was given by the Charm++ default static load balancer. After each iteration of the SP2 algorithm, the `GreedyCommLB` load balancer of Charm++ was called to migrate matrix and multiply chares in order to rebalance work and data. To demonstrate the efficiency of the `GreedyCommLB` load balancer, we show walltime vs. cores for the first iteration (only statically balanced), panels (a)-(c) of Fig. 4.4, and for the final iteration, panels (d)-(f) of Fig. 4.4. Notice that the difference in wall time between the first and last iteration is due to matrix fill-in (the decay slows from Fockian to density matrix). On the first iteration, we observe scaling roughly to $P = 30\ N$, corresponding to the default Charm++ data distribution. After a few iterations however, the communication aware persistence-based `GreedyCommLB` load balancer dynamically migrates chares to achieve a balance of very high quality. In applications, the persistence-based load balance will remain effective between SCF cycles, and also as atomic-positions gradually evolve, *e.g.* in a molecular dynamics simulation, geometry optimization, *etc.*, mitigating inefficiencies associated with the first iterations.

In Table 4.1, we list parameters for the fits to Amdahl's law, $T_s^\tau + T_p^\tau/p$, corresponding to the fitted lines in panels (d)-(f) of Fig. 4.4. Also given in Table 4.1 are the corresponding break-even core counts, $P_{\text{even}}^\tau = T_p^\tau/T_s^\tau$, the ratio between parallel

τ	$(H_2O)_N$	matrix	T_s [s]	T_p [s]	P_{even}
	30	750	1.95	1,981	1,016
10^{-10}	90	2,250	1.66	53,481	32,252
	150	3,750	2.74	224,825	81,909
	30	750	2.05	1,981	966
10^{-8}	90	2,250	1.41	46,459	32,898
	150	3,750	2.48	149,819	60,364
	30	750	1.80	1,621	899
10^{-6}	90	2,250	1.11	30,983	28,007
	150	3,750	2.55	65,284	25,553

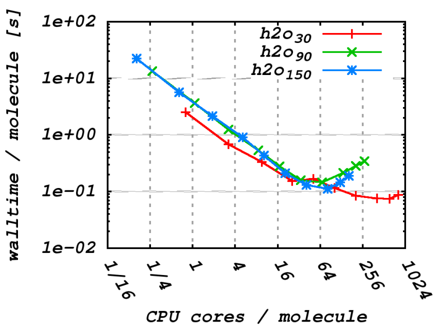
Table 4.1: Fit parameters for Amdahl's law, $T_s + T_p/P$, corresponding to the curves in panel (d)-(f) of Fig. 4.4. Matrix dimensions are shown in the third column labeled "matrix". Also listed is the break-even core-count $P_{\text{even}} = T_p/T_s$, providing a conservative estimate of parallel scalability.

and serial components. The break-even core count is a conservative estimate of the core count at which additional scaling becomes ineffective due the left-over serial component, which was found to be 1-3 seconds in all cases. Also, we notice a pronounced decrease in the parallel component with increasing values of τ, due to sparse-irregular effects. It should be noted that this analysis is a useful quantitative guide despite it being simplistic in ignoring more subtle scaling effects such as for example the scaling behavior of communication collectives and the details of network topology.

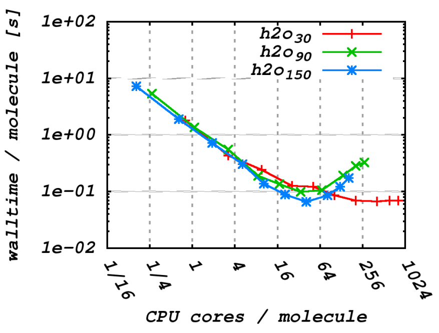
5. Conclusions. Relative to the ≈ 4 heavy atoms/core granularity achieved in the weak limit by advanced parallel methods [36], the default "static" distribution of work exhibited by our OpenMP/Charm++ implementation achieves roughly $P = 30\ N$, as shown in panels (a)-(c) of Fig. 4.4. Assuming 1 water molecule ≈ 2 heavy atoms, our default is $\sim 60\times$ more scalable. Once persistence is employed however, our results extend into the strong scaling regime, yielding $P = 400\ N$ to $600\ N$ as inferred from Table 4.1. For working accuracies and larger systems, *e.g.* $N \gg 150$ and $\tau \in \{10^{-8}, 10^{-12}\}$, we expect substantially better results as suggested by Table 4.1. We also expect substantially better results for problems with slower decay, as for example problems involving semi-conductors and metal oxides.

Based on the results given in Fig, the (very modest) serial component seems to be due to the Charm++ runtime. Larger calculations on larger computers will allow the reliable collection of diagnostics, as well as examination of the relationships between data locality and communication.

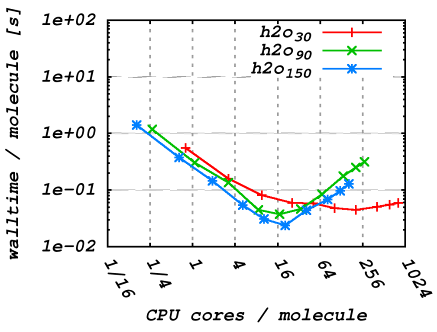
While our recursive, depth-first implementation of SpAMM with OpenMP exhibits good parallel scaling for larger chunk sizes, further improvements, including parallel performance at fine granularities, may require more explicit approaches to exploiting the inherent temporal and spatial localities present in SpAMM and to make contact with the deep memory cache hierarchy of the Magny-Cours architecture. Also, other computer platforms may not exhibit the pronounced non-uniform memory access (NUMA) effects common to the AMD Magny-Cours architecture, and we may expect the parallel performance of `SpAMM_omp` on those platforms to show improved scaling. Finally, it is known that the runtime has significant impact on the performance of SpAMM-like workloads [117], and other programming frameworks



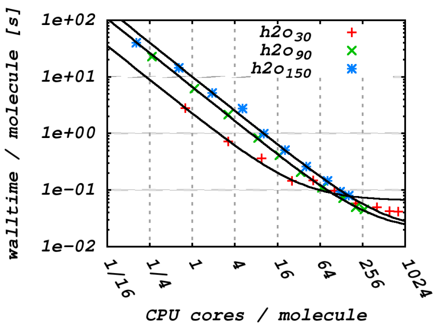
(a) First SP2 iteration, $\tau = 10^{-10}$.



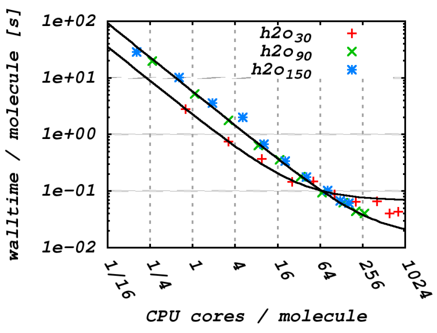
(b) First SP2 iteration, $\tau = 10^{-8}$.



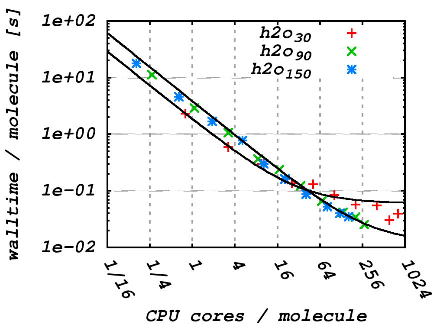
(c) First SP2 iteration, $\tau = 10^{-6}$.



(d) Last SP2 iteration. $\tau = 10^{-10}$.



(e) Last SP2 iteration, $\tau = 10^{-8}$.



(f) Last SP2 iteration. $\tau = 10^{-6}$.

Fig. 4.4: Shown in panels (a)-(c), scaling results of the first SP2 iteration under different thresholds. Shown in panels (d)-(f), scaling results of the last and fully load-balanced iteration (iteration 40) of SP2, under the same thresholds. As a guide, fits to Amdahl's law, $T_s + T_p/p$, are shown as solid lines, see Table 4.1 for fitting parameters.

might lead to improved parallel scaling.

These satisfactory results follow from over-decomposition of the three-dimensional convolution space, relative to conventional methods that involve decomposition in one or two dimensions, and from runtime systems that support the irregular task parallelism inherent in the generalized N-Body solvers framework. The ability to recursively generate singleton chares would greatly simplify the implementation of N-Body methods, and enhance their efficiency by eliminating the explicit management of tree-traversal as explained in Section 3.2. This prospect, together with a unified code base for N-Body solver collectives (based on established prototypes [51, 52, 53, 54, 55, 56, 48]), may offer a simple and well posed approach to meeting the challenges of increasing hardware complexity.

The software written for and used in this study is available online at `http://www.freeon.org/spammpack` [27], licensed under the terms of the BSD license [1].

6. Acknowledgments. NB and MC thank the LDRD program for funding this research under LDRD-ER grant 20110230ER. Both would also like to acknowledge generous support from the Ten-Bar Café providing stimulating and helpful discussions. This article was released under LA-UR-14-22050. The Los Alamos National Laboratory is operated by Los Alamos National Security, LLC for the NNSA of the USDoE under Contract No. DE-AC52- 06NA25396.

REFERENCES

[1] *BSD 3-Clause License*. `http://opensource.org/licenses/BSD-3-Clause`.
[2] *Charm++*. `http://charm.cs.uiuc.edu/`.
[3] *Concurrent Collections*. `https://software.intel.com/en-us/articles/intel-concurrent-collections-for-cc`.
[4] *CUDA Programming Guide*. `http://docs.nvidia.com/cuda/`.
[5] *Intel Cilk Plus*. `https://www.cilkplus.org/`.
[6] *Intel Threading Building Blocks*. `https://www.threadingbuildingblocks.org/`.
[7] *Nanos++*. `https://pm.bsc.es/nanox`.
[8] *Open Community Runtime*. `https://01.org/open-community-runtime`.
[9] *OpenCL*. `https://www.khronos.org/opencl/`.
[10] *OpenMP*. `http://openmp.org/`.
[11] *OpenUH: Open-source UH Compiler*. `http://web.cs.uh.edu/~openuh/`.
[12] *TASCEL*. `http://hpc.pnl.gov/tascel/`.
[13] *Wool - Fine Grained Independent Task Parallelism in C*.
[14] S. Kamal Abdali and David S. Wise, *Experiments with quadtree representation of matrices*, in Symbolic and Algebraic Computation, P. Gianni, ed., vol. 358 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 1989, pp. 96–108.
[15] Michael D. Adams and David S. Wise, *Seven at one stroke: results from a cache-oblivious paradigm for scalable matrix algorithms*, in Proceedings of the 2006 workshop on Memory system performance and correctness, MSPC '06, New York, NY, USA, 2006, ACM, pp. 41–50.
[16] R. R. Amossen and R. Pagh, *Faster join-projects and sparse matrix multiplications*, in Proceedings of the 12th International Conference on Database Theory, ACM, 2009, pp. 121–126.
[17] Grey Ballard, Aydin Buluç, James Demmel, Laura Grigori, Benjamin Lipshitz, Oded Schwartz, and Sivan Toledo, *Communication Optimal Parallel Multiplication of Sparse Random Matrices*, in Proceedings of the Twenty-fifth Annual ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13, New York, NY, USA, 2013, ACM, pp. 222–231.
[18] Axel D. Becke, *Density-Functional thermochemistry. III. The role of exact exchange*, The Journal of Chemical Physics, 98 (1993), pp. 5649–5652.
[19] Michele Benzi, Paolo Boito, and Nader Razouk, *Decay properties of spectral projectors with applications to electronic structure*, arXiv:math.NA, 1203.3953 (2012).
[20] ———, *Decay Properties of Spectral Projectors with Applications to Electronic Structure*, SIAM Review, 55 (2013), pp. 3–64.
[21] Michele Benzi and Gene H. Golub, *Bounds for the entries of matrix functions with applications to preconditioning*, BIT, 39 (1999), pp. 417–438.
[22] Michele Benzi and Nader Razouk, *Decay Bounds and $\mathcal{O}(n)$ Algorithms for Approximating Functions of Sparse Matrices*, Electron. T. Numer. Ana., 28 (2007), p. 16.

[23] Michele Benzi and Miroslav Tuma, *Orderings for factorized sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1851–1868.

[24] Abhinav Bhatele, Sameer Kumar, Chao Mei, James C. Phillips, Gengbin Zheng, and Laxmikant V. Kalé, *NAMD: A Portable and Highly Scalable Program for Biomolecular Simulations*, Tech. Report UIUCDCS-R-2009-3034, Department of Computer Science, University of Illinois at Urbana-Champaign, February 2009.

[25] Dario Bini and Grazia Lotti, *Stability of fast algorithms for matrix multiplication*, Numerische Mathematik, 36 (1980), pp. 63–72.

[26] Nicolas Bock, *Bug #445: Communication aware load balancers fail on chare arrays that are not full.*

[27] Nicolas Bock and Matt Challacombe, *spammpack, A High-Performance Implemenation of SpAMM*, 2011. `http://www.freeon.org/spammpack`.

[28] ———, *An Optimized Sparse Approximate Matrix Multiply for Matrices with Decay*, SIAM Journal on Scientific Computing, 35 (2013), pp. C72–C98.

[29] Nicolas Bock, Matt Challacombe, Chee Kwan Gan, Graeme Henkelman, Karoly Nemeth, Anders M. N. Niklasson, Anders Odell, Eric Schwegler, C. J. Tymczak, and Valery Weber, FreeON*: A suite of programs for linear scaling quantum chemistry*, 2011. `http://www.freeon.org/`.

[30] Urban Borštnik, Joost VandeVondele, Valéry Weber, and Jürg Hutter, *Sparse matrix multiplication: The distributed block-compressed sparse row library*, Parallel Computing, 40 (2014), pp. 47 – 58.

[31] D.R. Bowler and M.J. Gillan, *Density matrices in $\mathcal{O}(n)$ electronic structure calculations: theory and applications*, Computer Physics Communications, 120 (1999), pp. 95 – 108.

[32] David R. Bowler and T. Miyazaki, *Calculations for millions of atoms with density functional theory: linear scaling shows its potential*, Journal of Physics: Condensed Matter, 22 (2010), p. 074207.

[33] ———, *$\mathcal{O}(n)$ methods in electronic structure calculations*, Reports on Progress in Physics, 75 (2012), p. 036503.

[34] David R. Bowler, T. Miyazaki, and M. J. Gillan, *Parallel sparse matrix multiplication for linear scaling electronic structure calculations*, Computer Physics Communications, 137 (2001), pp. 255 – 273.

[35] ———, *Recent progress in linear scaling ab initio electronic structure techniques*, Journal of Physics: Condensed Matter, 14 (2002), p. 2781.

[36] David R. Bowler, Tsuyoshi Miyazaki, Lionel A. Truflandier, and Michael J. Gillan, *Comment on "Accurate and Scalable O(N) Algorithm for First-Principles Molecular-Dynamics Computations on Large Parallel Computers"*, arXiv:cont-mat.mtrl-sci, 1402.6828 (2014).

[37] V. Brázdová and David R. Bowler, *Automatic data distribution and load balancing with space-filling curves: implementation in CONQUEST*, Journal of Physics: Condensed Matter, 20 (2008), p. 275223.

[38] Aydin Buluç and John R. Gilbert, *Challenges and Advances in Parallel Sparse Matrix-Matrix Multiplication*, in ICPP '08: Proceedings of the 2008 37th International Conference on Parallel Processing, Washington, DC, USA, 2008, IEEE Computer Society, pp. 503–510.

[39] ———, *On the representation and multiplication of hypersparse matrices*, in 2008 IEEE International Symposium on Parallel and Distributed Processing, IEEE, Apr. 2008, pp. 1–11.

[40] ———, *On the representation and multiplication of hypersparse matrices*, in Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, April 2008, pp. 1–11.

[41] ———, *Highly parallel sparse matrix-matrix multiplication*, Technical Report UCSB-CS-2010-10, University of California, 2010.

[42] ———, *Parallel Sparse Matrix-Matrix Multiplication and Indexing: Implementation and Experiments*, Arxiv preprint arXiv:1109.3739, (2011).

[43] John C. Burant, Gustavo E. Scuseria, and Michael J. Frisch, *A linear scaling method for Hartree–Fock exchange calculations of large molecules*, The Journal of Chemical Physics, 105 (1996), pp. 8969–8972.

[44] Eric J Bylaska, Kevin Glass, Doug Baxter, Scott B Baden, and John H Weare, *Hard scaling challenges for ab initio molecular dynamics capabilities in NWChem: Using 100,000 CPUs per second*, Journal of Physics: Conference Series, 180 (2009), p. 012028.

[45] L. E. Cannon, *A Cellular Computer to Implement the Kaiman Filter Algorithm*, PhD thesis, Ph. D. Dissertation, Montana State University, 1969.

[46] Matt Challacombe, *A simplified density matrix minimization for linear scaling self-*

consistent field theory, J. Chem. Phys., 110 (1999), pp. 2332–2342.

[47] ———, *A general parallel sparse-blocked matrix multiply for linear scalingSCF theory.*, Comp. Phys. Comm., 128 (2000), p. 93.

[48] ———, *Linear scaling computation of the Fock matrix. V. hierarchical cubature for numerical integration of the exchange-correlation matrix*, J. Chem. Phys., 113 (2000), p. 10037.

[49] Matt Challacombe, *Linear Scaling Solution of the Time-Dependent Self-Consistent-Field Equations*, Computation, 2 (2014), pp. 1–11.

[50] Matt Challacombe and Nicolas Bock, *Fast Multiplication of Matrices with Decay*, arXiv:cs.DS, 1011.3534 (2010).

[51] ———, *An N-Body Solution to the Problem of Fock Exchange*, arXiv preprint arXiv:1401.6961, (2014).

[52] ———, *On the scalibility of Newton Schulz Iterations in an Approximate Linear Algebra*, in preparation, (2015).

[53] Matt Challacombe and Eric Schwegler, *Linear scaling computation of the Fock matrix*, J. Chem. Phys., 106 (1997), p. 5526.

[54] Matt Challacombe, Eric Schwegler, and Jan Almlöf, *Computational Chemistry: Review of Current Trends*, World Scientific, Singapore, 1996, pp. 53–107.

[55] ———, *Fast assembly of the Coulomb matrix: A quantum chemical tree code*, J. Chem. Phys., 104 (1996), pp. 4685–4698.

[56] ———, *Modern Developments in Hartree-Fock Theory: Fast Methods for Computing the Coulomb Matrix*, in Computational Chemistry: Reviews of Current Trends, J. Leszczynski, ed., vol. 1 of Computational Chemistry: Reviews of Current Trends, World Scientific, Singapore, 1996, pp. 53–107.

[57] Chen-Ting Chang, Yu-Sheng Chen, I-Wei Wu, and Jyh-Jiun Shann, *A Translation Framework for Automatic Translation of Annotated LLVM IR into OpenCL Kernel Function*, in Advances in Intelligent Systems and Applications - Volume 2, Jeng-Shyang Pan, Ching-Nung Yang, and Chia-Chen Lin, eds., vol. 21 of Smart Innovation, Systems and Technologies, Springer Berlin Heidelberg, 2013, pp. 627–636.

[58] Jie Chen and Edmond Chow, *A Newton-Schulz Variant for Improving the Initial Convergence in Matrix Sign Computation*, Preprint ANL/MCS-P5059-0114, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, 2014.

[59] Edmond Chow, Xing Liu, Mikhail Smelyanskiy, and Jeff R Hammond, *Parallel scalability of Hartree–Fock calculations*, The Journal of Chemical Physics, 142 (2015), p. 104103.

[60] Andrew D. Daniels, John M. Millam, and Gustavo E. Scuseria, *Semiempirical methods with conjugate gradient density matrix search to replace diagonalization for molecular systems containing thousands of atoms*, The Journal of Chemical Physics, 107 (1997), pp. 425–431.

[61] Murray S. Daw, *Model for energetics of solids based on the density matrix*, Phys. Rev. B, 47 (1993), pp. 10895–10898.

[62] Michel Daydé, Jack Dongarra, Vicente Hernández, and José M. L. M. Palma, eds., *High Performance Computing for Computational Science - VECPAR 2004*, vol. 3402 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004.

[63] S. Demko, W. F. Moss, and P. W. Smith, *Decay rates for inverses of band matrices*, Math. Comp., 43 (1984), pp. 491–499.

[64] Michael Driscoll, Evangelos Georganas, Penporn Koanantakool, Edgar Solomonik, and Katherine Yelick, *A Communication-Optimal N-Body Algorithm for Direct Interactions*, in Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on, IEEE, 2013, pp. 1075–1084.

[65] Jeremy D. Frens and David S. Wise, *Auto-blocking matrix-multiplication or tracking BLAS3 performance from source code*, SIGPLAN Not., 32 (1997), pp. 206–216.

[66] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, Andrea Dal Corso, Stefano de Gironcoli, Stefano Fabris, Guido Fratesi, Ralph Gebauer, Uwe Gerstmann, Christos Gougoussis, Anton Kokalj, Michele Lazzeri, Layla Martin-Samos, Nicola Marzari, Francesco Mauri, Riccardo Mazzarello, Stefano Paolini, Alfredo Pasquarello, Lorenzo Paulatto, Carlo Sbraccia, Sandro Scandolo, Gabriele Sclauzero, Ari P Seitsonen, Alexander Smogunov, Paolo Umari, and Renata M Wentzcovitch, *QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials*, Journal of Physics: Condensed Matter, 21 (2009), p. 395502.

[67] Filippo Gioachin, Pritish Jetley, Celso L. Mendes, Laxmikant V. Kalé, and Thomas R. Quinn, *Toward Petascale Cosmological Simulations with ChaNGa*, Tech.

Report 07-08, Parallel Programming Laboratory, Department of Computer Science, University of Illinois at Urbana-Champaign, 2007.

[68] J. N. Glosli, D. F. Richards, K. J. Caspersen, R. E. Rudd, J. A. Gunnels, and F. H. Streitz, *Extending Stability Beyond CPU Millennium: A Micron-scale Atomistic Simulation of Kelvin-Helmholtz Instability*, 2007. ACM Gordon Bell Prize.

[69] C. M. Goringe, E. Hernández, M. J. Gillan, and I. J. Bush, *Linear-scaling DFT-pseudopotential calculations on parallel computers*, Computer physics communications, 102 (1997), pp. 1–16.

[70] Peter Gottschling, David S. Wise, and Michael D. Adams, *Representation-transparent matrix algorithms with scalable performance*, in Proceedings of the 21st annual international conference on Supercomputing, ICS '07, New York, NY, USA, 2007, ACM, pp. 116–125.

[71] Alexander G. Gray and Andrew W. Moore, *N-Body Problems in Statistical Learning*, in Advances in Neural Information Processing Systems, vol. 4, MIT Press, 2001, pp. 521–527.

[72] Martyn F. Guest, Ian J. Bush, Huub J. J. Van Dam, Paul Sherwood, Jens M. H. Thomas, Joop H. Van Lenthe, Remco W. A. Havenith, and John Kendrick, *The GAMESS-UK electronic structure package: algorithms, developments and applications*, Molecular Physics, 103 (2005), pp. 719–747.

[73] Fred G. Gustavson, *Two Fast Algorithms for Sparse Matrices: Multiplication and Permuted Transposition*, ACM Trans. Math. Softw., 4 (1978), pp. 250–269.

[74] Jürgen Hafner, *Ab-initio simulations of materials using VASP: Density-functional theory and beyond*, Journal of Computational Chemistry, 29 (2008), pp. 2044–2078.

[75] T. Hamada, T. Narumi, R. Yokota, K. Yasuoka, K. Nitadori, and M. Taiji, *42 TFlops Hierarchical N-body Simulations on GPUs with Applications in Both Astrophysics and Turbulence*, 2009. ACM Gordon Bell Prize.

[76] William W. Hargrove, Forrest M. Hoffman, and Thomas Sterling, *The Do-It-Yourself Supercomputer*, Scientific American, 265 (2001), pp. 72–79.

[77] Peter Haynes, Chris Kriton Skylaris, Arash Mostofi, and Mike Payne, *ONETEP: Linear-scaling density-functional theory with plane waves*, 2010. `http://www2.tcm.phy.cam.ac.uk/onetep/`.

[78] Timothy G Heil, Stephen V O'Neil, and Henry F Schaefer, *High precision valence bond potential curve for the cl$_2$ molecule*, Chemical Physics Letters, 5 (1970), pp. 253–256.

[79] E. G. Hoel and H. Samet, *Data-Parallel Spatial Join Algorithms*, in Parallel Processing, 1994. ICPP 1994. International Conference on, vol. 3, aug. 1994, pp. 227 –234.

[80] P. Hohenberg and W. Kohn, *Inhomogeneous Electron Gas*, Phys. Rev., 136 (1964), pp. B864–B871.

[81] Arieh Iserles, *How Large is the Exponential of a Banded Matrix?*, J. New Zealand Math. Soc., 29 (2000), p. 177.

[82] T. Ishiyama, K. Nitadori, and J. Makino, *4.45 Pflops Astrophysical N-Body Simulation on K computer – The Gravitational Trillion-Body Problem*, 2012. ACM Gordon Bell Prize.

[83] Edwin H. Jacox and Hanan Samet, *Iterative spatial join*, ACM Trans. Database Syst., 28 (2003), pp. 230–256.

[84] Pritish Jetley, Filippo Gioachin, Celso Mendes, Laxmikant V Kale, and Thomas Quinn, *Massively parallel cosmological simulations with changa*, in Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, IEEE, 2008, pp. 1–12.

[85] Pritish Jetley, Filippo Gioachin, Celso Mendes, Laxmikant V. Kalé, and Thomas R. Quinn, *Massively parallel cosmological simulations with ChaNGa*, in Proceedings of IEEE International Parallel and Distributed Processing Symposium 2008, 2008.

[86] Laxmikant V. Kalé and Abhinav Bhatele, eds., *Parallel Science and Engineering Applications: The Charm++ Approach*, Taylor & Francis Group, CRC Press, Nov. 2013.

[87] Laxmikant V. Kale, Abhinav Bhatele, Eric J. Bohm, and James C. Phillips, *NAnoscale Molecular Dynamics (NAMD)*, in Encyclopedia of Parallel Computing (to appear), D. Padua, ed., Springer Verlag, 2011.

[88] Laxmikan V. Kalé, Sameer Kumar, and J. DeSouza, *A Malleable-Job System for Time-shared Parallel Machines*, in Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on, May 2002, pp. 230–230.

[89] Atsushi Kawai, Toshiyuki Fukushige, Junichiro Makino, and Makoto Taiji, *GRAPE-5: A special-purpose computer for N-body simulation*, arXiv preprint astro-ph/9909116, (1999).

[90] A. Kawai, T. Fushushige, and J. Makino, *Astrophysical N-Body Simulation*, 1999. ACM Gordon Bell Prize.

[91] W. Kohn and L. J. Sham, *Self-Consistent Equations Including Exchange and Correlation Effects*, Phys. Rev., 140 (1965), pp. A1133–A1138.

[92] Dongryeol Lee and Alexander G. Gray, *Faster Gaussian Summation: Theory and Experiment*, in Proceedings of the Twenty-second Conference on Uncertainty in Artificial Intelligence, 2006.

[93] ———, *Fast High-dimensional Kernel Summations Using the Monte Carlo Multipole Method*, in Advances in Neural Information Processing Systems (NIPS) 21 (Dec 2008), MIT Press, 2009.

[94] X.-P. Li, R. W. Nunes, and David Vanderbilt, *Density-matrix electronic-structure method with linear system-size scaling*, Phys. Rev. B, 47 (1993), pp. 10891–10894.

[95] Michael D. Lieberman, Jagan Sankaranarayanan, and Hanan Samet, *A Fast Similarity Join Algorithm Using Graphics Processing Units*, in Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, Washington, DC, USA, 2008, IEEE Computer Society, pp. 1111–1120.

[96] Jonathan Lifflander, Sriram Krishnamoorthy, and Laxmikant Kalé, *Steal Tree: Low-Overhead Tracing of Work Stealing Schedulers*, in Proceedings of the 34rd ACM SIGPLAN conference on Programming Language Design and Implementation, (To Appear), PLDI '13, ACM, 2013.

[97] K. Patrick Lorton and David S. Wise, *Analyzing block locality in morton-order and morton-hybrid matrices*, in Proceedings of the 2006 workshop on Memory performance: Dealing with Applications, systems and architectures, MEDEA '06, New York, NY, USA, 2006, ACM, pp. 5–12.

[98] W. Ma and S. Krishnamoorthy, *Data-driven fault tolerance for work stealing computations*, in Proceedings of the 26th ACM international conference on Supercomputing, ACM, 2012, pp. 79–90.

[99] Junichiro Makino, Toshiyuki Fukushige, Masaki Koga, and Ken Namura, *GRAPE-6: The massively-parallel special-purpose computer for astrophysical particle simulation*, arXiv preprint astro-ph/0310702, (2003).

[100] Junichiro Makino and Makoto Taiji, *Scientific Simulations with Special-Purpose Computers–the GRAPE Systems*, vol. 1, 1998.

[101] Junichiro Makino, Makoto Taiji, Toshikazu Ebisuzaki, and Daiichiro Sugimoto, *Grape-4: A massively parallel special-purpose computer for collisional n-body simulations*, The Astrophysical Journal, 480 (1997), p. 432.

[102] Gabriel Martinez, Mark Gardner, and Wu-chun Feng, *CU2CL: A CUDA-to-OpenCL translator for multi-and many-core architectures*, in Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on, IEEE, 2011, pp. 300–307.

[103] R. McWeeny, *The Density Matrix in Self-Consistent Field Theory. I. Iterative Construction of the Density Matrix*, P. Roy. Soc. Lond A Mat., 235 (1956), pp. 496–509.

[104] Chao Mei, Yanhua Sun, Gengbin Zheng, Eric J. Bohm, Laxmikant V. Kalé, James C.Phillips, and Chris Harrison, *Enabling and Scaling Biomolecular Simulations of 100 Million Atoms on Petascale Machines with a Multicore-optimized Message-driven Runtime*, in Proceedings of the 2011 ACM/IEEE conference on Supercomputing, Seattle, WA, November 2011.

[105] Harshitha Menon and Laxmikant Kalé, *A Distributed Dynamic Load Balancer for Iterative Applications*, in Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis, SC '13, New York, NY, USA, 2013, ACM, pp. 15:1–15:11.

[106] John M. Millam and Gustavo E. Scuseria, *Linear scaling conjugate gradient density matrix search as an alternative to diagonalization for first principles electronic structure calculations*, The Journal of Chemical Physics, 106 (1997), pp. 5569–5577.

[107] ———, *Linear scaling conjugate gradient density matrix search as an alternative to diagonalization for first principles electronic structure calculations*, The Journal of Chemical Physics, 106 (1997), pp. 5569–5577.

[108] Priti Mishra and Margaret H. Eich, *Join processing in relational databases*, ACM Comput. Surv., 24 (1992), pp. 63–113.

[109] Tsuyoshi Miyazaki, *Ultra-large-scale first-principles calculations by the k computer*, NIMS NOW, 11 (2014).

[110] A. Moore, A. Connolly, C. Genovese, Alexander G. Gray, L. Grone, N. Kanidoris, R. Nichol, J. Schneider, A. Szalay, I. Szapudi, and L. Wasserman, *Fast Algorithms and Efficient Statistics: n-point Correlation Functions*, in Proceedings of MPA/MPE/ESO Conference Mining the Sky, 2000.

[111] T. Narumi, Y. Ohno, N. Okimoto, T. Koishi, A. Suenaga, N. Futatsugi, R. Yanai,

R. Himeno, S. Fujikawa, M. Taiji, and M. Ikei, *A 55 TFLOPS Simulation of Amyloid-forming Peptides from Yeast Prion Sup35 with the Special-purpose Computer System MDGRAPE-3*, 2006. ACM Gordon Bell Prize.

[112] Anders M. N. Niklasson, *Expansion algorithm for the density matrix*, Phys. Rev. B, 66 (2002), p. 5.

[113] Anders M. N. Niklasson, C. J. Tymczak, and Matt Challacombe, *Trace resetting density matrix purification in O(N) self-consistent-field theory*, J. Chem. Phys., 118 (2003), pp. 8611–8620.

[114] Gabriel Noaje, Christophe Jaillet, and Michaël Krajecki, *Source-to-source code translator: OpenMP C to CUDA*, in High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on, IEEE, 2011, pp. 512–519.

[115] Christian Ochsenfeld, Christopher A. White, and Martin Head-Gordon, *Linear and sublinear scaling formation of Hartree–Fock-type exchange matrices*, The Journal of Chemical Physics, 109 (1998), pp. 1663–1669.

[116] Adam H. R. Palser and David E. Manolopoulos, *Canonical purification of the density matrix in electronic-structure theory*, Phys. Rev. B, 58 (1998), pp. 12704–12711.

[117] Artur Podobas, Mats Brorsson, and Karl-filip Faxén, *A comparative performance study of common and popular task-centric programming frameworks*, (2015), pp. 1–28.

[118] A. Rahimian, I. Lashuk, S. Veerapaneni, S. A. Chandramowlishwaran, J. Vetter, R. Vuduc, D. Zorin, and G. Biros, *Petascale Direct Numerical Simulation of Blood Flow on 200K Cores and Heterogeneous Architectures*, 2010. ACM Gordon Bell Prize.

[119] Ioan Raicu, Ian T. Foster, and Pete Beckman, *Making a Case for Distributed File Systems at Exascale*, in Proceedings of the Third International Workshop on Large-scale System and Application Performance, LSAP '11, New York, NY, USA, 2011, ACM, pp. 11–18.

[120] Parikshit Ram, Dongryeol Lee, William March, and Alexander G. Gray, *Linear-time Algorithms for Pairwise Statistical Problems*, in Advances in Neural Information Processing Systems (NIPS) 22 (Dec 2009), MIT Press, 2010.

[121] Amit Sabne, Putt Sakdhnagool, and Rudolf Eigenmann, *Effects of compiler optimizations in OpenMP to CUDA translation*, in OpenMP in a Heterogeneous World, Springer, 2012, pp. 169–181.

[122] Hanan Samet, *The design and analysis of spatial data structures*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[123] ———, *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann, 2006.

[124] Osman Sarood, Akhil Langer, Abhishek Gupta, and Laxmikant Kale, *Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget*, in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '14, Piscataway, NJ, USA, 2014, IEEE Press, pp. 807–818.

[125] Osman Sarood, Esteban Meneses, and L. V. Kale, *A "Cool" Way of Improving the Reliability of HPC Machines*, in Proceedings of The International Conference for High Performance Computing, Networking, Storage and Analysis, Denver, CO, USA, November 2013.

[126] Eric Schwegler, Matt Challacombe, and Martin Head-Gordon, *Linear scaling computation of the Fock matrix. II. Rigorous bounds on exchange integrals and incremental fock build*, J. Chem. Phys., 106 (1997), pp. 9708–9717.

[127] Dimitrios Skarlatos, Polyvios Pratikakis, and Dionisios Pnevmatikatos12, *Towards Reliable Task Parallel Programs*, in 5th HiPEAC Workshop on Design for Reliability, 2013.

[128] Chris Kriton Skylaris, Peter D. Haynes, Arash A. Mostofi, and Mike C. Payne, *Introducing ONETEP: Linear-scaling density functional simulations on parallel computers*, The Journal of Chemical Physics, 122 (2005), p. 084119.

[129] Thomas Sterling, Donald J. Becker, Daniel Savarese, John E. Dorband, Udaya A. Ranawake, and Charles V. Packer, *Beowulf: A Parallel Workstation For Scientific Computation*, in In Proceedings of the 24th International Conference on Parallel Processing, CRC Press, 1995, pp. 11–14.

[130] A. Szabo and N.S. Ostlund, *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*, Dover Publications, 1996.

[131] M. Valiev, E.J. Bylaska, N. Govind, K. Kowalski, T.P. Straatsma, H.J.J. Van Dam, D. Wang, J. Nieplocha, E. Apra, T.L. Windus, and W.A. de Jong, *NWChem: A comprehensive and scalable open-source solution for large scale molecular simulations*, Computer Physics Communications, 181 (2010), pp. 1477 – 1489.

[132] Robert A. van de Geijn and J. Watts, *SUMMA: Scalable universal matrix multiplication algorithm*, Concurrency Practice and Experience, 9 (1997), pp. 255–274.

[133] Joost VandeVondele, Urban Borštnik, and Jürg Hutter, *Linear Scaling Self-Consistent Field Calculations with Millions of Atoms in the Condensed Phase*, J. Chem. Theory Comput., 8 (2012), pp. 3565–3573.

[134] John Paul N. Walters, *Fault-tolerant Techniques for High Performance Computing and a Bioinformatics Application*, PhD thesis, Detroit, MI, USA, 2007. AAI3295992.

[135] Michael S. Warren, *2HOT: an improved parallel hashed oct-tree n-body algorithm for cosmological simulation*, in Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, 2013, p. 72.

[136] Michael S. Warren and John K. Salmon, *Astrophysical N-Body Simulations Using Hierarchical Tree Data Structures*, in Supercomputing '92, Los Alamitos, 1992, IEEE Comp. Soc., pp. 570–576. (1992 Gordon Bell Prize winner).

[137] ———, *Simulation of 9 million gravitating stars by parallelizing a tree code*, 1992. ACM Gordon Bell Prize.

[138] ———, *A parallel, portable and versatile treecode*, SIAM, Philadelphia, 1995, ch. 1.

[139] ———, *Simulating the motion of 322,000,000 self-gravitating particles*, 1997. ACM Gordon Bell Prize.

[140] Michael S. Warren, John K. Salmon, D. J. Becker, M. P. Goda, and T. Sterling, *Two problems: vortex fluid flow modeled with 360,000 particles; galaxy formation following 10,000,000 selfgravitating particles*, 1997. ACM Gordon Bell Prize.

[141] David S. Wise, *Representing matrices as quadtrees for parallel processors: extended abstract*, SIGSAM Bull., 18 (1984), pp. 24–25.

[142] ———, *Ahnentafel Indexing into Morton-Ordered Arrays, or Matrix Locality for Free*, in Euro-Par 2000 Parallel Processing, Arndt Bode, Thomas Ludwig, Wolfgang Karl, and Roland Wismüller, eds., vol. 1900 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2000, pp. 774–783.

[143] David S. Wise and John Franco, *Costs of quadtree representation of nondense matrices*, Journal of Parallel and Distributed Computing, 9 (1990), pp. 282 – 296.

[144] David S. Wise, Jeremy D. Frens, Yuhong Gu, and Gregory A. Alexander, *Language support for Morton-order matrices*, SIGPLAN Not., 36 (2001), pp. 24–33.

[145] Mitsuo Yokokawa, Fumiyoshi Shoji, Atsuya Uno, Motoyoshi Kurokawa, and Tadashi Watanabe, *The K computer: Japanese next-generation supercomputer development project*, in ISLPED '11: Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design, Piscataway, NJ, USA, 2011, IEEE Press.

[146] Gongpu Zhao, Juan R. Perilla, Ernest L. Yufenyuy, Xin Meng, Bo Chen, Jiying Ning, Jinwoo Ahn, Angela M. Gronenborn, Klaus Schulten, Christopher Aiken, and Peijun Zhang, *Mature HIV-1 capsid structure by cryo-electron microscopy and all-atom molecular dynamics*, Nature, 497 (2013), pp. 643–646.

[147] Gengbin Zheng, Xiang Ni, and Laxmikant V Kalé, *A scalable double in-memory checkpoint and restart scheme towards exascale*, in Dependable Systems and Networks Workshops (DSN-W), 2012 IEEE/IFIP 42nd International Conference on, IEEE, 2012, pp. 1–6.